

1 **CLAIMS**

2

3 1. A system comprising:

4 a source computing device to generate an encrypted directory name based
5 on a plaintext name that conforms to a syntax; and

6 a recipient computing device, coupled to the source computing device, to
7 receive the encrypted directory name, to verify that the encrypted directory name
8 is an encryption of a plaintext name that conforms to the syntax without
9 decrypting the encrypted directory name, and to verify that the directory name is
10 an encryption of a plaintext name that is not a duplicative name without
11 decrypting the encrypted directory name.

12

13 2. A system as recited in claim 1, wherein the source computing device
14 and the recipient computing device together implement a serverless distributed file
15 system.

16

17 3. A system as recited in claim 1, wherein the source computing device
18 is to generate the encrypted directory name by:

19 receiving a plaintext name;

20 generating, based on the plaintext name, a mapped name;

21 encoding the mapped name; and

22 encrypting the encoded name.

1 4. A system as recited in claim 3, further comprising:

2 generating, based on the mapped name, a declassified name and
3 corresponding case information;

4 wherein the encoding comprises encoding the declassified name; and

5 wherein the encrypting comprises encrypting both the encoded declassified
6 name and the case information.

7
8 5. A system as recited in claim 3, wherein the generating comprises
9 generating the mapped name only if the received name is syntactically legal.

10
11 6. A system as recited in claim 3, wherein the encoding comprises
12 encoding the mapped name only if the received name is syntactically legal.

13
14 7. A system as recited in claim 3, wherein generating the mapped name
15 comprises:

16 checking whether the identifier is equal to one of a plurality of illegal
17 names;

18 if the name is not equal to one of the plurality of illegal names, then
19 checking whether the name is equal to one of the plurality of illegal names
20 followed by one or more particular characters;

21 if the name is not equal to one of the plurality of illegal names followed by
22 one or more particular characters, then using the name as the mapped name; and

23 if the identifier is equal to one of the plurality of illegal names followed by
24 one or more particular characters, then using as the mapped name the name with
25 one of the particular characters removed.

1
2 8. A system as recited in claim 7, wherein the particular character
3 comprises an underscore.

4
5 9. A system as recited in claim 3, wherein encoding the mapped name
6 comprises:

7 reversing the order of characters in the mapped name;

8 removing, from the reversed name, all trailing characters of a particular
9 type;

10 initializing the encoded name with a string of one bits equal in number to a
11 number of trailing characters removed from the reversed name followed by a zero
12 bit;

13 selecting a first character from the reversed name;

14 encoding the first character using a first coding table;

15 adding, to the encoded name, a series of zero bits followed by the encoded
16 first character;

17 for each additional character in the reversed name,

18 selecting the next character in the reversed name,

19 encoding the next character using a second coding table,

20 adding, to the encoded name, a series of zero bits followed by the
21 encoded next character; and

22 removing any trailing zero bits and the one bit preceding the trailing zero
23 bits from the encoded name.

1 **10.** A system as recited in claim 9, wherein the characters of a particular
2 type are the characters that are coded to zero using the first coding table.

3
4 **11.** A system as recited in claim 9, wherein the first coding table and the
5 second coding table are Huffman coding tables.

6
7 **12.** A system as recited in claim 9, wherein each coding in the first
8 coding table is the same as a corresponding coding in the second coding table, but
9 the second coding table codes additional characters not coded by the first coding
10 table.

11
12 **13.** A system as recited in claim 9, wherein for the first character and
13 each additional character, encoding the character only if a set of leading bits of the
14 character are zero, and further comprising adding the character to the encoded
15 name if the set of leading bits of the character are not zero.

16
17 **14.** A system as recited in claim 3, wherein encoding the mapped name
18 comprises:

19 reversing the order of characters in the mapped name;

20 removing, from the reversed name, all trailing characters of a particular
21 type;

22 initializing the encoded name with a string of one bits equal in number to a
23 number of trailing characters removed from the reversed name followed by a zero
24 bit;

25 selecting a first character from the reversed name;

1 encoding the first character using a first coding table;

2 adding, to the encoded name, a series of zero bits followed by the encoded

3 first character;

4 for each additional character in the reversed name,

5 selecting the next character in the reversed name,

6 encoding the next character using one of a plurality of additional

7 coding tables,

8 adding, to the encoded name, a series of zero bits followed by the

9 encoded next character; and

10 removing any trailing zero bits and the one bit preceding the trailing zero

11 bits from the encoded name.

12

13 **15.** A system as recited in claim 3, wherein encrypting the encoded

14 identifier comprises using a block cipher to encrypt the encoded identifier.

15

16 **16.** A system as recited in claim 3, wherein encrypting the encoded

17 identifier comprises using cipher block chaining to encrypt the encoded identifier.

18

19 **17.** A system as recited in claim 1, wherein the recipient computing

20 device is to verify that the encrypted directory name conforms to the syntax by

21 checking whether a first block of the encrypted directory name is zero, and

22 determining that the encrypted directory name conforms to the syntax if the first

23 block is not equal to zero.

1 **18.** A system as recited in claim 1, wherein the recipient computing
2 device is to verify that the directory name is not a duplicative name by comparing
3 the encrypted directory name to a plurality of other encrypted directory names,
4 checking whether the encrypted directory name is the same as any of the other
5 encrypted directory name, and determining that the encrypted directory name is
6 not a duplicative name if the encrypted directory name is not the same as any of
7 the plurality of encrypted directory names.

8

9 **19.** A method comprising:
10 receiving an identifier;
11 generating, based on the identifier, a mapped identifier;
12 encoding the mapped identifier; and
13 encrypting the encoded identifier.

14

15 **20.** A method as recited in claim 19, wherein the identifier comprises
16 one of: a file name, a folder name, and a directory name.

17

18 **21.** A method as recited in claim 19, further comprising:
19 generating, based on the mapped identifier, a declassified identifier and
20 corresponding case information;
21 wherein the encoding comprises encoding the declassified identifier; and
22 wherein the encrypting comprises encrypting both the encoded declassified
23 identifier and the case information.

1 **22.** A method as recited in claim 21, wherein generating the declassified
2 identifier and corresponding case information comprises:

3 for each character that has both an upper-case and a lower-case form,
4 storing the character in upper-case form and recording in the case information
5 whether the character was in upper-case form or lower-case form.

6
7 **23.** A method as recited in claim 22, further comprising:

8 storing the character in upper-case form only if the character is one of
9 particular set of characters; and

10 storing the character without altering its case if the character is not one of
11 the particular set of characters.

12
13 **24.** A method as recited in claim 23, wherein the particular set of
14 characters comprises the extended ASCII character set.

15
16 **25.** A method as recited in claim 19, wherein the generating comprises
17 generating the mapped identifier only if the received identifier is syntactically
18 legal.

19
20 **26.** A method as recited in claim 19, wherein the encoding comprises
21 encoding the mapped identifier only if the received identifier is syntactically legal.

22
23 **27.** A method as recited in claim 19, further comprising:
24 receiving an encrypted identifier from another device;
25 decrypting the encrypted identifier;

1 decoding the decrypted identifier; and
2 demapping the decoded decrypted identifier.

3
4 **28.** A method as recited in claim 27, further comprising:
5 receiving encrypted case information corresponding to the encrypted
6 identifier;

7 decrypting the case information;
8 recasifying, using the decrypted case information, the decrypted identifier;
9 and
10 wherein the demapping comprises demapping the recasified decoded
11 decrypted identifier.

12
13 **29.** A method as recited in claim 19, wherein generating the mapped
14 identifier comprises:

15 checking whether the identifier is equal to one of a plurality of illegal
16 identifiers;

17 if the identifier is not equal to one of the plurality of illegal identifiers, then
18 checking whether the identifier is equal to one of the plurality of illegal identifiers
19 followed by one or more particular characters;

20 if the identifier is not equal to one of the plurality of illegal identifiers
21 followed by one or more particular characters, then using the identifier as the
22 mapped identifier; and

23 if the identifier is equal to one of the plurality of illegal identifiers followed
24 by one or more particular characters, then using as the mapped identifier the
25 identifier with one of the particular characters removed.

1
2 **30.** A method as recited in claim 29, wherein the particular character
3 comprises an underscore.

4
5 **31.** A method as recited in claim 19, wherein encoding the mapped
6 identifier comprises:

7 reversing the order of characters in the mapped identifier;
8 removing, from the reversed identifier, all trailing characters of a particular
9 type;

10 initializing the encoded identifier with a string of one bits equal in number
11 to a number of trailing characters removed form the reversed identifier followed
12 by a zero bit;

13 selecting a first character from the reversed identifier;
14 encoding the first character using a first coding table;
15 adding, to the encoded identifier, a series of zero bits followed by the
16 encoded first character;

17 for each additional character in the reversed identifier,
18 selecting the next character in the reversed identifier,
19 encoding the next character using a second coding table,
20 adding, to the encoded identifier, a series of zero bits followed by
21 the encoded next character; and

22 removing any trailing zero bits and the one bit preceding the trailing zero
23 bits from the encoded identifier.

1 **32.** A method as recited in claim 31, wherein the characters of a
2 particular type are the characters that are coded to zero using the first coding table.

3
4 **33.** A method as recited in claim 31, wherein the first coding table and
5 the second coding table are Huffman coding tables.

6
7 **34.** A method as recited in claim 31, wherein each coding in the first
8 coding table is the same as a corresponding coding in the second coding table, but
9 the second coding table codes additional characters not coded by the first coding
10 table.

11
12 **35.** A method as recited in claim 31, wherein for the first character and
13 each additional character, encoding the character only if a set of leading bits of the
14 character are zero, and further comprising adding the character to the encoded
15 identifier if the set of leading bits of the character are not zero.

16
17 **36.** A method as recited in claim 19, wherein encoding the mapped
18 identifier comprises:

19 reversing the order of characters in the mapped identifier;

20 removing, from the reversed identifier, all trailing characters of a particular
21 type;

22 initializing the encoded identifier with a string of one bits equal in number
23 to a number of trailing characters removed from the reversed identifier followed
24 by a zero bit;

25 selecting a first character from the reversed identifier;

1 encoding the first character using a first coding table;

2 adding, to the encoded identifier, a series of zero bits followed by the

3 encoded first character;

4 for each additional character in the reversed identifier,

5 selecting the next character in the reversed identifier,

6 encoding the next character using one of a plurality of additional

7 coding tables,

8 adding, to the encoded identifier, a series of zero bits followed by

9 the encoded next character; and

10 removing any trailing zero bits and the one bit preceding the trailing zero

11 bits from the encoded identifier.

12

13 **37.** A method as recited in claim 19, wherein encrypting the encoded

14 identifier comprises using a block cipher to encrypt the encoded identifier.

15

16 **38.** A system as recited in claim 19, wherein encrypting the encoded

17 identifier comprises using cipher block chaining to encrypt the encoded identifier.

18

19 **39.** A system as recited in claim 19, wherein the encrypting comprises

20 encrypting the encoded identifier to generate, using a block cipher, a ciphertext

21 having a fixed size.

1 **40.** A system as recited in claim 39, further comprising indicating that
2 the received identifier cannot be encrypted if the length of the encoded identifier
3 exceeds the fixed size by more than one.

4

5 **41.** One or more computer-readable memories containing a computer
6 program that is executable by a processor to perform the method recited in claim
7 19.

8

9 **42.** A method comprising:
10 receiving an encrypted identifier;
11 verifying, without decrypting the encrypted identifier, that the encrypted
12 identifier is an encryption of another identifier that conforms to a syntax; and
13 verifying, without decrypting the encrypted identifier, that the encrypted
14 identifier is not an encryption of the same other identifier as one or more other
15 encrypted identifiers.

16

17 **43.** A method as recited in claim 42, wherein verifying that the
18 encrypted identifier is an encryption of another identifier that conforms to the
19 syntax comprises:

20 checking whether a first block of the encrypted identifier is zero;
21 determining that the encrypted directory name conforms to the syntax if the
22 first block is not equal to zero; and
23 determining that the encrypted directory name does not conform to the
24 syntax if the first block is equal to zero.

1 **44.** A method as recited in claim 42, wherein verifying that the
2 encrypted identifier is not an encryption of the same other identifier as one or
3 more other encrypted identifiers comprises:

4 comparing the encrypted identifier to the one or more other encrypted
5 identifiers; and

6 determining that the encrypted identifier is the same as one or more other
7 encrypted identifiers if the comparing indicates that the encrypted identifier is
8 equal to one of the other encrypted identifiers.

9
10 **45.** One or more computer-readable memories containing a computer
11 program that is executable by a processor to perform the method recited in claim
12 42.

13
14 **46.** A system comprising:

15 a plurality of encrypted identifiers;

16 a syntax verifier to determine whether a newly received encrypted identifier
17 is an encryption of a legal name without decrypting the newly received encrypted
18 identifier; and

19 a duplication identifier to determine whether the newly received encrypted
20 identifier is an encryption of the same name as any of the plurality of encrypted
21 identifiers without decrypting either the newly received encrypted identifier or any
22 of the plurality of encrypted identifiers.

1 **47.** One or more computer-readable media having stored thereon a
2 plurality of instructions that, when executed by one or more processors of a
3 computer, causes the one or more processors to perform acts including:

4 receiving a plaintext identifier;

5 generating a ciphertext by encrypting the plaintext identifier only if the
6 plaintext identifier is syntactically legal; and

7 wherein the encrypting allows another device to verify, without decrypting
8 the ciphertext, that the plaintext identifier is not identical to another plaintext
9 identifier maintained by the other device.

10
11 **48.** One or more computer-readable media as recited in claim 47,
12 wherein generating the ciphertext comprises:

13 generating, based on the plaintext identifier, a mapped identifier;

14 encoding the mapped identifier; and

15 encrypting the encoded identifier.

16
17 **49.** One or more computer-readable media as recited in claim 48,
18 wherein generating the ciphertext further comprises:

19 generating, based on the mapped identifier, a declassified identifier and
20 corresponding case information;

21 wherein the encoding comprises encoding the declassified identifier; and

22 wherein the encrypting comprises encrypting both the encoded declassified
23 identifier and the case information.

1 **50.** One or more computer-readable media as recited in claim 48,
2 wherein generating the mapped identifier comprises:

3 checking whether the plaintext identifier is equal to one of a plurality of
4 illegal identifiers;

5 if the plaintext identifier is not equal to one of the plurality of illegal
6 identifiers, then checking whether the plaintext identifier is equal to one of the
7 plurality of illegal identifiers followed by one or more particular characters;

8 if the plaintext identifier is not equal to one of the plurality of illegal
9 identifiers followed by one or more particular characters, then using the plaintext
10 identifier as the mapped identifier; and

11 if the plaintext identifier is equal to one of the plurality of illegal identifiers
12 followed by one or more particular characters, then using as the mapped identifier
13 the plaintext identifier with one of the particular characters removed.

14
15 **51.** One or more computer-readable media as recited in claim 48,
16 wherein encoding the mapped identifier comprises:

17 reversing the order of characters in the mapped identifier;

18 removing, from the reversed identifier, all trailing characters of a particular
19 type;

20 initializing the encoded identifier with a string of one bits equal in number
21 to a number of trailing characters removed from the reversed identifier followed
22 by a zero bit;

23 selecting a first character from the reversed identifier;

24 encoding the first character using a first coding table;

1 adding, to the encoded identifier, a series of zero bits followed by the
2 encoded first character;
3 for each additional character in the reversed identifier,
4 selecting the next character in the reversed identifier,
5 encoding the next character using a second coding table,
6 adding, to the encoded identifier, a series of zero bits followed by
7 the encoded next character; and
8 removing any trailing zero bits and the one bit preceding the trailing zero
9 bits from the encoded identifier.

10
11 **52.** One or more computer-readable media as recited in claim 51,
12 wherein each coding in the first coding table is the same as a corresponding coding
13 in the second coding table, but the second coding table codes additional characters
14 not coded by the first coding table.

15
16 **53.** One or more computer-readable media as recited in claim 48,
17 wherein encoding the mapped identifier comprises:

18 reversing the order of characters in the mapped identifier;
19 removing, from the reversed identifier, all trailing characters of a particular
20 type;
21 initializing the encoded identifier with a string of one bits equal in number
22 to a number of trailing characters removed from the reversed identifier followed
23 by a zero bit;
24 selecting a first character from the reversed identifier;
25 encoding the first character using a first coding table;

1 adding, to the encoded identifier, a series of zero bits followed by the
2 encoded first character;

3 for each additional character in the reversed identifier,

4 selecting the next character in the reversed identifier,

5 encoding the next character using one of a plurality of additional
6 coding tables,

7 adding, to the encoded identifier, a series of zero bits followed by
8 the encoded next character; and

9 removing any trailing zero bits and the one bit preceding the trailing zero
10 bits from the encoded identifier.

11
12 **54.** One or more computer-readable media as recited in claim 48,
13 wherein encrypting the encoded identifier comprises using a block cipher to
14 encrypted the encoded identifier.

15
16 **55.** A method comprising:

17 receiving an encrypted identifier;

18 receiving encrypted case information corresponding to the encrypted
19 identifier;

20 decrypting the encrypted identifier;

21 decrypting the case information;

22 decoding the decrypted identifier;

23 recasifying, using the decrypted case information, the decrypted identifier;

24 and

25 demapping the recasified decoded decrypted identifier.

1
2 **56.** One or more computer-readable memories containing a computer
3 program that is executable by a processor to perform the method recited in claim
4 55.

5
6 **57.** A method implemented at a computing device, the method
7 comprising:

8 receiving a directory entry that is encrypted, wherein the computing device
9 does not have a key needed for decrypting the directory entry;

10 verifying that the directory entry is an encryption of a syntactically legal
11 name; and

12 verifying that the directory entry is not an encryption of the same name as
13 any other directory entry maintained by the computer device.

14
15 **58.** One or more computer-readable memories containing a computer
16 program that is executable by a processor to perform the method recited in claim
17 57.

18
19 **59.** A system comprising:

20 a plurality of encrypted directory entries;

21 a syntax verifier to determine whether a new encrypted directory entry is an
22 encryption of a legal name without decrypting the new encrypted directory entry;
23 and

24 a duplication identifier to determine whether the new encrypted directory
25 entry is an encryption of the same name as any of the plurality of encrypted

1 directory entries without decrypting either the new encrypted directory entry or
2 any of the plurality of encrypted directory entries.

3
4 **60.** One or more computer-readable media having stored thereon a
5 plurality of instructions that, when executed by one or more processors of a
6 computer, causes the one or more processors to perform acts including:

7 receiving a plaintext directory entry;
8 verifying that the plaintext directory entry is syntactically legal;
9 encrypting the plaintext directory entry only if the plaintext directory entry
10 is syntactically legal;
11 communicating the encrypted directory entry to another device; and
12 wherein the encrypting allows the other device to verify, without
13 decrypting the encrypted directory entry, that the directory entry is not identical to
14 any other directory entry maintained by the other device.

15
16 **61.** One or more computer-readable media as recited in claim 60,
17 wherein the computer is part of a serverless distributed file system.

18
19 **62.** One or more computer-readable media as recited in claim 60,
20 wherein the plaintext directory entry comprises a file name.

21
22 **63.** One or more computer-readable media as recited in claim 60,
23 wherein the plaintext directory entry comprises a directory name.

1 **64.** One or more computer-readable media as recited in claim 60,
2 wherein the plaintext directory entry comprises a folder name.

3
4 **65.** One or more computer-readable media as recited in claim 60,
5 wherein the plurality of instructions further cause the one or more processors to
6 perform acts including:

7 receiving an encrypted directory entry from another device;
8 decrypting the encrypted directory entry;
9 decoding the decrypted identifier; and
10 demapping the decoded decrypted identifier.

11
12 **66.** One or more computer-readable media as recited in claim 65, further
13 comprising:

14 receiving encrypted case information corresponding to the encrypted
15 directory entry;
16 decrypting the case information;
17 recasifying, using the decrypted case information, the decrypted identifier;
18 and
19 wherein the demapping comprises demapping the recasified decoded
20 decrypted identifier.

21
22 **67.** One or more computer-readable media as recited in claim 60,
23 wherein encrypting the plaintext directory entry comprises:

24 generating, based on the plaintext directory entry, a mapped identifier;
25 encoding the mapped identifier; and

1 encrypting the encoded identifier.

2

3 **68.** One or more computer-readable media as recited in claim 67, further

4 comprising indicating that the received plaintext directory entry cannot be

5 encrypted if the length of the encoded identifier exceeds a fixed encrypted

6 directory entry size by more than one.

7

8 **69.** One or more computer-readable media as recited in claim 67,

9 wherein encrypting the plaintext directory entry further comprises:

10 generating, based on the mapped identifier, a declassified identifier and

11 corresponding case information;

12 wherein the encoding comprises encoding the declassified identifier; and

13 wherein the encrypting comprises encrypting both the encoded declassified

14 identifier and the case information.

15

16 **70.** One or more computer-readable media as recited in claim 67,

17 wherein generating the mapped identifier comprises generating the mapped

18 identifier only if the received plaintext directory entry is syntactically legal.

19

20 **71.** One or more computer-readable media as recited in claim 67,

21 wherein the encoding comprises encoding the mapped identifier only if the

22 received plaintext directory entry is syntactically legal.

1 **72.** One or more computer-readable media as recited in claim 67,
2 wherein generating the mapped identifier comprises:

3 checking whether the plaintext directory entry is equal to one of a plurality
4 of illegal identifiers;

5 if the plaintext directory entry is not equal to one of the plurality of illegal
6 identifiers, then checking whether the plaintext directory entry is equal to one of
7 the plurality of illegal identifiers followed by one or more particular characters;

8 if the plaintext directory entry is not equal to one of the plurality of illegal
9 identifiers followed by one or more particular characters, then using the plaintext
10 directory entry as the mapped identifier; and

11 if the plaintext directory entry is equal to one of the plurality of illegal
12 identifiers followed by one or more particular characters, then using as the mapped
13 identifier the plaintext directory entry with one of the particular characters
14 removed.

15
16 **73.** One or more computer-readable media as recited in claim 72,
17 wherein the particular character comprises an underscore.

18
19 **74.** One or more computer-readable media as recited in claim 67,
20 wherein encoding the mapped identifier comprises:

21 reversing the order of characters in the mapped identifier;

22 removing, from the reversed identifier, all trailing characters of a particular
23 type;

1 initializing the encoded identifier with a string of one bits equal in number
2 to a number of trailing characters removed from the reversed identifier followed
3 by a zero bit;

4 selecting a first character from the reversed identifier;

5 encoding the first character using a first coding table;

6 adding, to the encoded identifier, a series of zero bits followed by the
7 encoded first character;

8 for each additional character in the reversed identifier,

9 selecting the next character in the reversed identifier,

10 encoding the next character using a second coding table,

11 adding, to the encoded identifier, a series of zero bits followed by
12 the encoded next character; and

13 removing any trailing zero bits and the one bit preceding the trailing zero
14 bits from the encoded identifier.

15
16 75. One or more computer-readable media as recited in claim 74,
17 wherein each coding in the first coding table is the same as a corresponding coding
18 in the second coding table, but the second coding table codes additional characters
19 not coded by the first coding table.

20
21 76. One or more computer-readable media as recited in claim 74,
22 wherein the characters of a particular type are the characters that are coded to zero
23 using the first coding table.

1 77. One or more computer-readable media as recited in claim 74,
2 wherein the first coding table and the second coding table are Huffman coding
3 tables.

4

5 78. One or more computer-readable media as recited in claim 74,
6 wherein each coding in the first coding table is the same as a corresponding coding
7 in the second coding table, but the second coding table codes additional characters
8 not coded by the first coding table.

9

10 79. One or more computer-readable media as recited in claim 74,
11 wherein for the first character and each additional character, encoding the
12 character only if a set of leading bits of the character are zero, and further
13 comprising adding the character to the encoded identifier if the set of leading bits
14 of the character are not zero.

15

16 80. One or more computer-readable media as recited in claim 67,
17 wherein encrypting the encoded identifier comprises using a block cipher to
18 encrypt the encoded identifier.

19

20 81. One or more computer-readable media as recited in claim 60,
21 wherein the encrypting further comprises generating, using a block cipher, the
22 encrypted directory entry having a fixed size.

1 **82.** A computing device comprising:

2 a client component to encrypt only directory entries that are syntactically
3 legal, and to encrypt the directory entries in a manner that allows another device to
4 verify, without decrypting the encrypted entries, that the directory entries are not
5 identical to any other directory entries maintained by the other device; and

6 a server component to receive encrypted directory entries, to verify that the
7 received encrypted directory entries are encryptions of syntactically legal directory
8 entries, and to verify that the received encrypted directory entries are not
9 encryptions of directory entries identical to any other directory entries maintained
10 by the device.

11
12 **83.** A computing device as recited in claim 82, wherein the server
13 component can receive directory entries encrypted by the client component of the
14 computing device as well as client components of other computing devices.

15
16 **84.** A system comprising:

17 a server component;

18 a client component coupled to the server component; and

19 wherein the server component and the client component together ensure
20 that multiple entries in a directory cannot have the same name, that all entries in
21 the directory are syntactically legal, and that the server component does not have
22 access to the unencrypted names of entries in the directory.

1 **85.** A system as recited in claim 84, wherein the server component and
2 the client component are implemented on two different computing devices.

3
4 **86.** A system as recited in claim 84, wherein each of the server
5 component and the client component comprise one or more software modules.

6
7 **87.** A system comprising:

8 means for verifying that a plaintext directory entry is syntactically legal;

9 means for encrypting the plaintext directory entry only if the plaintext
10 directory entry is syntactically legal;

11 means for communicating the encrypted directory entry to another device;

12 and

13 wherein the encrypting allows the other device to verify, without
14 decrypting the encrypted directory entry, that the directory entry is not identical to
15 any other directory entry maintained by the other device.